14　整列法比較

あみだくじを使って単純整列法の挿入法と泡法について原理を学び，比較回数と移動回数を減らす改良をしてきました。あみだくじではデータが少なかったので，もっと大量のデータで比較してみましょう。

14.1　新規プログラム

14.1.1　名前

ユニット名　　SortingU
プロジェクト名　SortingP

14.1.2　Form のプロパティ

Name	FormSorting
Caption	整列法比較（学生証番号　氏名）
Position	poDesktopCenter

14.1.3　メインパネルと終了ボタン

いつも通り。

14.1.4　データ構造

```
const
  NoMax    = 400;
  DataMax  = 600;
  PenWidth = 2;
type
  TDataNo = 0..NoMax;
  TData   = array [TDataNo] of Integer;            // 0 番は作業用
  TFormSorting = class(TForm)
    PanelMain: TPanel;

    途中省略

  private
    Data0     : TData;                             // 整列前のデータ
    Data      : TData;                             // 整列中のデータ
```

14.1.5　Form の OnCreate ハンドラー

```
procedure TFormSorting.FormCreate(Sender: TObject);
begin
  ClientWidth  := NoMax*PenWidth+PanelMain.Width;
  ClientHeight := DataMax;
```

```
  Canvas.Pen.Width := PenWidth;
  NewData;
end; (* FormCreate *)
```

14.1.6　メソッド追加

```
(********************** フォームのメソッド ********************)
procedure TFormSorting.DrawOne(No : TDataNo);
          (* No 番目のデータを描く *)
begin
  with Canvas do
    begin
      Pen.Color := clLime;                    // 前のデータを消す
      MoveTo(No*2,DataMax);
      LineTo(No*2,0);
      Pen.Color := clAqua;                    // 今のデータを描く
      LineTo(No*2,Data[No]);
    end;
end; (* DrawOne *)

procedure TFormSorting.NewData;
          (* 新しいデータを作る *)
var
  No : TDataNo;
begin
  for No := 1 to NoMax do              // 元のデータをでたらめに作る
    Data0[No] := Random(DataMax);
  StandBy;
end; (* NewData *)

procedure TFormSorting.StandBy;
          (* 整列の準備をする *)
var
  No : TDataNo;
begin
  Data := Data0;                       // 元のデータに初期設定する
  for No := 1 to NoMax do              // 全データを描く
    DrawOne(No);
end; (* StandBy *)
```

14.1.7　実行

フォームのサイズが自動的に設定されます。データを表示したはずですが見えません。Image の Canvas と違って Form の Canvas は絵が保存されないので，他の画面の影に隠れたりすると，消えてしまうのです。

14.1.8　整列データ変更

Button をメインパネルの中，終了ボタンの少し上に置く。

Name	ButtonNew
Caption	新データ

OnClick ハンドラー

```
procedure TFormSorting.ButtonNewClick(Sender: TObject);
begin
  NewData;
end; (* ButtonNewClick *)
```

14.1.9　実行

新データボタンを押すと，新しいデータを作成して表示します。表示は，データを水色の線分で表して大小関係が視覚的にわかるようにしたものです。
これを長さの順に並べ替えます。

14.2　整列

14.2.1　整列法選択ラジオグループ

RadioGroup をメインパネルの中に置く。

Align	alTop
Name	RadioGroupSort
Caption	整列法
Items	バブルソート（原始版）
	バブルソート（改良版）
	インサートソート（原始版）
	インサートソート（改良版）
	シェルソート
	クイックソート

OnClick ハンドラー

```
procedure TFormSorting.RadioGroupSortClick(Sender: TObject);
begin
  StandBy;
end; (* RadioGroupSortClick *)
```

14.2.2 実行

新データボタンを押さなくても，整列法を選ぶと現在のデータを表示して待機します。

14.2.3 実行ボタン

Button をメインパネルの中に置く。

Name	ButtonExecute
Caption	実行

OnClick ハンドラー

```
procedure TFormSorting.ButtonExecuteClick(Sender: TObject);
begin
  StandBy;
  case RadioGroupSort.ItemIndex of
    0 : BubbleSort0(1,NoMax);
    1 : BubbleSort(1,NoMax);
    (*
    2 : InsertSort0(1,NoMax);
    3 : InsertSort(1,NoMax);
    4 : ShellSort(1,NoMax);
    5 : QuickSort(1,NoMax);
    *)
  end;
end; (* ButtonExecuteClick *)
```

14.2.4 Form のメソッド

```
function  TFormSorting.Chiisai(No1,No2 : TDataNo) : Boolean;
          (* No1 番目が No2 番目より小さい *)
begin
  Chiisai := Data[No1] < Data[No2];
end; (* Chiisai *)

procedure TFormSorting.Idou(No1,No2 : TDataNo);
          (* No1 番目を No2 番目に移動する *)
begin
  Data[No2] := Data[No1];
  DrawOne(No2);
end; (* Idou *)

procedure TFormSorting.Koukan(No1,No2 : TDataNo);
          (* No1 番目と No2 番目を交換する *)
          (* 0 番目を退避用に使う         *)
begin
  Idou(No1,0);
  Idou(No2,No1);
  Idou(0,No2);
end; (* Koukan *)
```

```
(******************** 整列法 ********************)
procedure TFormSorting.BubbleSort0(IMin,IMax : TDataNo);
          (* 泡整列法　(原始版)　  *)
var
  IEnd,I : TDataNo;
begin
  for IEnd := IMax-1 downto IMin do
    begin
      for I := IMin to IEnd do
        if Chiisai(I,I+1)
          then Koukan(I,I+1);
    end;
end; (* BubbleSort0 *)

procedure TFormSorting.BubbleSort(IMin,IMax : TDataNo);
          (* 泡整列法　(改良版)　  *)
var
  IEnd,I : TDataNo;
  IStart,IFirst,ILast : TDataNo;
begin
  IStart := IMin;
  IEnd := IMax-1;
  repeat
    IFirst := IMax;
    ILast := IMin;
    Idou(IStart,0);
    for I := IStart to IEnd do
      if Chiisai(0,I+1)
        then begin
               Idou(I+1,I);
               ILast := I;
               if I < IFirst
                 then IFirst := I;
             end
        else begin
               Idou(0,I);
               Idou(I+1,0);
             end;
    Idou(0,IEnd+1);
    if IFirst > IMin
      then IStart := IFirst-1;
    IEnd := ILast-1;
  until IStart > IEnd;
end; (* BubbleSort *)
```

14.2.5　実行

泡整列法で整列すると，右のほうから順に確定していくとともに，左のまだ確定しない部分も不完全ながら大まかに整列していることがわかります。

原始版と比べて改良版はかなりスピードアップしていることがわかります。

14.3　統計量

スピードアップしていることはわかりますが，統計データを調べて数値的に表しましょう。

14.3.1　表示パネル

メインパネルの中に Panel を 3 つ置く。

Align	alTop	alTop	alTop
Name	PanelHikaku	PanelIdou	PanelTime
Caption	比較回数	移動回数	所要時間

14.3.2　変数追加

```
    HikakuSuu : Integer;                                    // 比較回数
    IdouSuu   : Integer;                                    // 移動回数
    ByouSuu   : Real;                                       // 所要時間
```

14.3.3　イベントハンドラー変更

```
procedure TFormSorting.ButtonExecuteClick(Sender: TObject);
var
  TimeStart,TimeEnd : TDateTime;
begin
  StandBy;
  IdouSuu := 0;
  HikakuSuu := 0;
  TimeStart := Now;
  case RadioGroupSort.ItemIndex of
    0 : BubbleSort0(1,NoMax);
    1 : BubbleSort(1,NoMax);

    2 : InsertSort0(1,NoMax);
    3 : InsertSort(1,NoMax);
    4 : ShellSort(1,NoMax);
    5 : QuickSort(1,NoMax);

  end;
  TimeEnd := Now;
  ByouSuu := (TimeEnd-TimeStart)*24*60*60;
  PanelIdou.Caption := Format('移動回数%10d 回',[IdouSuu]);
  PanelHikaku.Caption := Format('比較回数%10d 回',[HikakuSuu]);
  PanelJikan.Caption := Format('所要時間%10.2f 秒',[ByouSuu]);
end; (* ButtonExecuteClick *)
```

Now は基準の日の午前 0 時から現在までの時間を返す関数で，単位は日です。整数部分が日数を表し，小数部分が今日の午前 0 時から現在までの時間を表します。午後 3 時は $15/24 = 0.625$，午後 4 時 48 分は $(16 * 60 + 48)/(24 * 60) = 0.7$ となります。

14.3.4　メソッド変更

```
procedure TFormSorting.Idou(No1,No2 : TDataNo);
          (* No1 番目を No2 番目に移動する *)
begin
  Data[No2] := Data[No1];
  DrawOne(No2);
  Inc(IdouSuu);
end; (* Idou *)

function  TFormSorting.Chiisai(No1,No2 : TDataNo) : Boolean;
          (* No1 番目が No2 番目より小さい *)
begin
  Chiisai := Data[No1] < Data[No2];
  Inc(HikakuSuu);
end; (* Chiisai *)
```

14.3.5　実行

14.4　挿入整列法

挿入整列法の原始版（改良してないもの）と改良版（比較も移動も不要なものをなくしたもの）を追加します。

subsubsection 問題

実行部分を完成させなさい。

```
procedure TFormSorting.InsertSort0(IMin,IMax : TDataNo);
          (* 挿入整列法　(原始版)　 *)
var
  IStart,I : TDataNo;
begin

(* 4行 *)

end; (* InsertSort0 *)

procedure TFormSorting.InsertSort(IMin,IMax : TDataNo);
          (* 挿入整列法　(改良版)　 *)
var
  IStart,I : TDataNo;
begin

(* 11行 *)

end; (* InsertSort *)
```

14.4.1　実行

泡整列法と比較してどうですか。

14.5　シェル整列法

シェル整列法を追加します。

14.5.1　問題

実行部分を完成させなさい。

注意

前回紹介した ShellSort では,

KawaStart を 1〜TateSuu-Arasa とし
Kawa 番目と Kawa+Arasa 番目を比較する

ようにしたので,

Dec(Kawa,Arasa)

を実行したとき，Kawa が負になることがあり，TKawaNo の範囲外になってしまいます。
そのため，Kawa の型を TKawaNo ではなく Integer としています。

しかし

KawaStart を Arasa+1〜TateSuu とし
Kawa-Arasa 番目と Kawa 番目を比較する

ようにすれば，Kawa が TKawaNo の範囲内で処理できます。

このように完成させなさい。

```
procedure TFormSorting.ShellSort(IMin,IMax : TDataNo);
          (* シェル整列法 *)
const
  Bairitu = 0.67;
var
  IStart,I : TDataNo;
  Arasa    : TDataNo;
begin
  Arasa := IMax-IMin;
  repeat

(* １２行 *)

  until Arasa = 1;
end; (* ShellSort *)
```

14.5.2　実行

早いですね。

14.6　負荷をかける

早すぎて振る舞いがよくわからないので，動作を遅くします。ただし無意味に遅くするのではなく，比較回数と移動回数が変わるたびに表示させることで遅くします。

14.6.1　表示か非表示か

RadioGroup をメインパネルの中に置く。

Align	alTop
Name	RadioGroupHyouji
Caption	途中経過表示
Items	表示 (遅い)
	非表示 (早い)
ItemIndex	1

14.6.2　変数追加

Form の private 部に追加。

```
Hyouji :  Boolean;
```

14.6.3　ButtonExecuteClick ハンドラーに追加

実行部の先頭に追加。

```
Hyouji := RadioGroupHyouji.ItemIndex = 0;
```

14.6.4　メソッド Idou と Chiisai に追加

それぞれの実行部の最後に追加

```
  if Hyouji
    then begin
           PanelIdou.Caption := Format('移動回数%10d 回',[IdouSuu]);
           PanelIdou.Repaint;
         end;
end; (* Idou *)

  if Hyouji
    then begin
           PanelHikaku.Caption := Format('比較回数%10d 回',[HikakuSuu]);
           PanelHikaku.Repaint;
         end;
end; (* Chiisai *)
```

14.6.5　実行

負荷をかけて，挿入法改良版とシェル法を比較しなさい。

14.7　クイックソート

高速整列法の代表として，クイックソートを紹介します。

14.7.1　原理

整列したいデータが与えられたら，大きいデータと小さいデータの 2 つのグループに分類します。大きい方を左に小さい方を右に置きます。それぞれのグループについて，また大きいデータと小さいデータの 2 つの小グループに分類します。これを繰り返し行って，すべてのグループが 1 つのデータだけになったとき，整列されています。

大きいデータ，小さいデータは何を基準にして決めるのでしょうか。ちょうど半々に分類されると理想的ですが，特殊なデータでないとそれは無理です。

ここでは各グループの左端のデータを基準にして，それより大きいか小さいかで分類することにします。

14.7.2　グループ分け

(1) グループの左端のデータを基準として取り出す。取り出した後を穴という。

(2) 穴の右から始めて右端まで順々に調べていく。

(3) 基準より小さいときはそのままにして右に進む。

(4) 基準より大きいときは穴に移す。今までの穴の右のデータをそこに移す。今までの穴の右が新しい穴になる。

(5) 右端まで調べ終わったら，基準にしたデータを穴に戻す。

例

	31	41	59	26	53	58	97	93	23	84	元のデータ
31		41	59	26	53	58	97	93	23	84	先頭を基準に選ぶ
31	41		59	26	53	58	97	93	23	84	基準より大きい，穴へ
31	41	59		26	53	58	97	93	23	84	基準より大きい，穴へ
31	41	59		26	53	58	97	93	23	84	基準より小さい，そのまま
31	41	59	53		26	58	97	93	23	84	基準より大きい，穴へ
31	41	59	53	58		26	97	93	23	84	基準より大きい，穴へ
31	41	59	53	58	97		26	93	23	84	基準より大きい，穴へ
31	41	59	53	58	97	93		26	23	84	基準より大きい，穴へ
31	41	59	53	58	97	93		26	23	84	基準より小さい，そのまま
31	41	59	53	58	97	93	84		23	26	基準より大きい，穴へ
	41	59	53	58	97	93	84	㉛	23	26	基準を穴へ

グループ分け（split）が終わったとき，基準になった 31 は “正しい位置” に納まっています。同様のグループ分けを 41〜84 と 23〜26 について再帰的に行います。

14.7.3　再帰プロシージャ QuickSort

```
procedure TFormSorting.QuickSort(IMin,IMax : TDataNo);
          (* クィックソート 再帰的 *)
var
  Hole : TDataNo;

  procedure Split;
            (* Data[IMin] より大きいものと小さいものに分ける *)
  var
    No : TDataNo;
  begin
    Idou(IMin,0);                   // 左端を基準として取り出す
    Hole := IMin;                   // 取り出した後を穴という
    for No := IMin+1 to IMax do     // 穴の右から右端まで調べる
      begin
        if Chiisai(0,No)            // 基準より大きいとき
          then begin
                 Idou(No,Hole);     // 穴に移し
                 Idou(Hole+1,No);   // 穴の右をその後に移す
                 Inc(Hole);         // 穴の右が新しい穴になる
               end
      end;
    Idou(0,Hole);                   // 基準を穴に戻す
  end;  Split

begin
  Split;                            // 大小２つのグループに分ける
  if Hole > IMin+1                  // 左グループに２個以上あるとき
    then QuickSort(IMin,Hole-1);
  if Hole < IMax-1                  // 右グループに２個以上あるとき
    then QuickSort(Hole+1,IMax);
end; (* QuickSort *)
```

14.7.4　実行

負荷をかけて振る舞いを見ましょう。